UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/418,943	10/15/1999	TIMOTHY CHARLES SOWELL	202231	9508

7590 06/12/2009

MARK JOY
LEYDIG VOIT & MAYER LTD
TWO PRUDENTIAL PLAZA SUITE 4900
180 NORTH STETSON
CHICAGO, IL 606016780

EXAMINER	
NGUYEN, NGA B	

ART UNIT	PAPER NUMBER
3692	

MAIL DATE	DELIVERY MODE
06/12/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

RECORD OF ORAL HEARING

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex Parte TIMOTHY CHARLES SOWELL

Appeal 2009-000451
Application No. 09/418,943
Technology Center 3600

Oral Hearing Held: May 14, 2009

Before HUBERT C. LORIN, LINDA E. HORNER, and ANTON W. FETTING, Administrative Patent Judges

ON BEHALF OF THE APPELLANT:

MARK JOY, ESQ.
Leydig, Voit & Mayer, Ltd.
Two Prudential Plaza, Suite 4900
180 North Stetson
Chicago, IL 60601-6780

The above-entitled matter came on for hearing on Thursday, May 14, 2009, commencing at 9:32 a.m., at the U.S. Patent and Trademark Office, 600 Dulany Street, Alexandria, Virginia, before Jan Jablonsky, Notary Public.

JUDGE LORIN: Again, this is Judge Lorin speaking, and the panel also has Judges Horner and Fetting.

Counsel, this is Appeal No. 2009-0451. We have reviewed the record. You have 20 minutes. You may begin.

MR. JOY: Okay.

Honorable members of the Board, as I said earlier, my name is Mark Joy, and I represent the applicant, Timothy Charles Sowell, and the assignee, Invent Systems, Inc., and this is with regard to Application No. 09/418,943, filed on October 5, 1999.

What I intend to do is initially describe the claimed invention, which is probably pretty easy to understand. I don't think there's anything particularly difficult to understand in the wording itself, but I'll still go over it just briefly. I'll then briefly describe the teachings of the cited references, and then address particular points raised in the briefs.

In general -- and I also recognize that there's -- there might be some additional issues raised by changes in law that have happened since the initial briefing of this, but I assume that would be handled separately. My intention today is just to address the obviousness rejections that have been raised in the final office action.

In general, I believe, as I said before, the appellant's briefs are clear with regard to the shortcomings of the obviousness rejections of the presently pending claims. Combined teachings of Archibald, Knapton, and Sobeski do not teach each of the recited elements of the independent claims, nor does there appear to be, for reasons I'll state later, any reason to modify

Archibald according to the teachings, especially, of Sobeski in a way that would result in the presently claimed invention, and in particular, the last clause of the independent claims that state that software usage is measured according to object instances -- at least one object class.

In particular, I'd like to address Sobeski.

First of all, I don't disagree with the final office action and the answer's characterization of Archibald. I think the main difference is in Applicant's and the Patent Office's interpretation of the effect of Sobeski's teaching on Archibald.

In particular, I think that Figure 3 of Sobeski and its associated written description, rather than teaching charging a customer according to monitored customer uses of software, teaches a license manager that performs a gateway/authorization function to ensure that current usage of software is within use limits imposed by a license file associated with the software, and it's our belief that nowhere does Sobeski disclose using the license manager and license file to monitor customer usage of software for the purpose of charging customers for that monitored use, and this results -- we believe that the combination of Sobeski with the teachings of Archibald and Knapton would result in a authorization confirmation functionality being added to Archibald, rather than Applicant's claimed charging -- quote, "charging the customer according to use of distributed software modules," and then I skipped a little bit, "wherein software use is measured according to object instances created from the distributed software modules."

Just generally addressing the claims themselves, the appeal claims are generally directed to methods and the software module structure that facilitate charging customers for use of software modules distributed to customer sites.

Claims 53 to 55 are directed to a particular arrangement of program and data within the software module that facilitates executing the claimed use-based methods for charging customers for using distributed software containing an object from which instances are rendered.

JUDGE HORNER: Mr. Joy, if -- if Archibald teaches charging for use --

MR. JOY: Yes.

JUDGE HORNER: -- of a software module and Knapton teaches object oriented programming as known --

MR. JOY: Uh-huh.

JUDGE HORNER: -- in the art and Sobeski teaches monitoring for purposes of licensing at the object level, why wouldn't it have been obvious to extend Archibald, in terms of charging for us, to the next level down, so to speak, in the program and look at charging per object?

MR. JOY: And that's actually the reason why I wanted to do this oral hearing. I feel very strongly that the teachings of -- of -- I'm sorry -- Sobeski, at columns 5 and 6, which discuss the license file, 202, and the license manager, 204, are very unequivocal in the sense that they are performing a gateway function.

You know, they're -- they're -- they're not monitoring the use of the software. What they're doing is they're -- they're performing an authorization function up front -- in other words, not even letting the person run the software if it isn't authorized, and therefore, the combination of Archibald with Sobeski teaches an authorization function with regard to objects, object instances, rather than monitoring and charging the use of that software based upon the creation of -- of instances from an object class.

JUDGE HORNER: But Sobeski does talk about a usage tag of the license file.

So, it does look at usage in terms of --

MR. JOY: Absolutely. In fact, when I first saw that, I had to take a second look at that, and if you look at the usage that they're talking about there, it's talking about -- I just read it again this morning. Where is it?

JUDGE HORNER: I think -- I'm looking at the bottom of column 7.

MR. JOY: Yes. Right. The usage tag is actually telling -- is part of this authorization function that says are you -- is the user allowed to use it in run time or are they allowed to use it at design time, and that's it.

It's just another part of this threshold decision-making process that's carried out in Sobeski. It says, is this person going to use this object in an editor, or is this person going to use it at a -- at run time?

JUDGE HORNER: But I guess my -- my question is, let's assume for a moment that Sobeski only teaches the authorization piece.

MR. JOY: Uh-huh.

JUDGE HORNER: But it's teaching authorization not at a program level but at an object level, and it's talking about licenses, and licenses you typically pay for. So, now you're looking at licensing at an object level, and you already know from Archibald that you can monitor usage of a program to charge per use.

So, why wouldn't the combined teachings of those two lead somebody of ordinary skill to charge for the license by usage based on the teaching in Archibald at the object level as taught in Sobeski?

MR. JOY: I'm probably going to just sound redundant to what I mentioned before. I believe that, you know, that goes beyond what the teaching of Sobeski teaches. I mean, it's a pre-authorization scheme --

JUDGE HORNER: But we're looking at -- I'm sorry. We're looking at the combined teachings, though, not just Sobeski.

MR. JOY: Oh, absolutely.

JUDGE HORNER: Okay.

MR. JOY: And I don't think that -- you know, the -- reading the -- if you read the background of our application and -- in particular, the background, I think -- the important aspect of this is that, in the particular environment, which is -- this is industrial process control, but we have not limited -- I am not suggesting that we're limiting it.

What they -- what they -- what they did is they -- they said, you know, there's the -- there's a problem with -- you know, people buy software and -- and we have no way of knowing what the true value is of that software.

So, what we're going to do is, rather than set a fixed price for buying the software, we're going to monitor its usage, and how are we going to monitor that usage? We're going to do it by this license manager which detects the creation of objects from a base class, and I don't think that either -- Archibald, as the Examiner stated in the final office action and in the answer -- they agree that there's -- that sort of scheme just is not present in Archibald, and then if you go on and look at Sobeski, I think that the teachings of Sobeski are that, you know, even if you're doing it in an object situation, that it's looking at it at the beginning, whether or not, you know, they're allowed to use it in the first place, rather than saying use it as many times as you want, we're going to charge you ever time you use it, every time you create a new object.

JUDGE FETTING: Well, doesn't -- I mean, doesn't Archibald at least give one of ordinary skill the idea that every time you run an object called a program, that you're going to charge for it? I mean, if -- if they're running -- in Archibald, if they're going to run that program 10 times, are they not going to get charged 10 times?

MR. JOY: I'm not sure that it's based upon running. It's been a while since I read the -- read it in detail, Archibald, but I think it was more a matter of every time you've created a copy of the program.

JUDGE FETTING: Every time you run the program, you make a copy.

MR. JOY: I'll have to take your word for it on that. I'm not sure. But I don't disagree that it's a use-based -- Archibald is -- I mean, I've never disagreed with that. Archibald is a use-based software billing scheme.

My distinction, what I believe is patentable, is the way in which we carry ours out. I don't think that Archibald -- I'm sorry -- that Sobeski or Knapton suggest the particular way in which the -- the customer is charged and the way it's monitored, the usage is monitored, as mentioned in the latter portion of the independent claims.

JUDGE FETTING: I don't think anybody is suggesting that Sobeski does. I think what the Examiner and we are suggesting is that Archibald has discussed the basic method of charging.

What Sobeski does is give one of ordinary skill the insight that you don't have to do it just at the program level, that you can do it in the sub-program level, as well, because Sobeski shows one of ordinary skill that you can actually monitor each of the instances that's happening at the sub-program level, but the incentive to do the charging would still come from Archibald.

So, I guess the question is, given the -- given Sobeski's insight, wouldn't one of ordinary skill, wanting to make a buck, just gather from Archibald, hey, there's a whole 'nother level that we can be charging at?

MR. JOY: I don't believe it -- it would teach someone of ordinary skill in the art to do it based upon the creation of object instances. I think that -- I think that Sobeski is a reference that teaches that there are objects in

this world, but more that there is a verification that can occur at the object level.

It's a -- it's a pre-authorization. In fact, I've got -- there's -- like independent claim 41, I believe, specifically calls out and distinguishes authorization -- I'm sorry, it's claim 37 -- and distinguishes authorizing as opposed to the actual measurement of -- of usage at a later time.

Sobeski, I think, is pretty clear in -- in what it believes this -- its functionality is used for, and that's just to verify that a user is authorized to use particular software objects, rather than monitoring the usage of those software objects and charging accordingly.

JUDGE LORIN: Counsel, I would like to get some of the terms clarified.

MR. JOY: Okay.

JUDGE LORIN: This is Judge Lorin.

MR. JOY: Okay.

JUDGE LORIN: You mentioned software modules in the claim, object, and instances.

MR. JOY: Okay.

JUDGE LORIN: Now, am I understanding correctly that you agree that Archibald teaches a user-based system for -- that monitors the usage of the software modules?

MR. JOY: I would agree. The modules are -- are generic enough. It's basically just saying a package of software.

JUDGE LORIN: Okay.

Do you -- do you further agree that it suggests a user-based system for -- for charging -- the use of software that monitors the usage of objects?

MR. JOY: Archibald? No.

JUDGE LORIN: It doesn't go to objects.

MR. JOY: No.

JUDGE LORIN: Your position is it goes to the software module level but doesn't go further to the objects.

MR. JOY: That's correct. And there's more to it. I mean, there's an object class. The software module includes an object, object class, and instances are created from that object class.

So, there's a -- there's another feature involved there, and that's the -- there's an object factory, basically, that -- that creates those objects, and at the time of creation, there's a -- there's a monitoring function which monitors the number of objects that are created.

JUDGE LORIN: So, what you're saying is, even though Archibald is applicable to software modules, your -- yours is applicable to software modules that -- that involve -- that are more specific, that involve object classes.

MR. JOY: Yes. And also that we define and claim a particular way of monitoring the usage based upon the creation of those instances.

JUDGE LORIN: Right. I understand that.

MR. JOY: Okay.

JUDGE LORIN: I just wanted to get these terms clarified --

MR. JOY: Yes.

JUDGE LORIN: -- you know, to see where the differences are.

MR. JOY: Right. Okay. I would say the difference, you know, in a nutshell, is with regard to the manner -- the manner in which usage is monitored, and that's -- monitored and thereafter charged, and that's based upon the creation of object instances from a previously downloaded or distributed object class, which is contained within a software module.

JUDGE LORIN: Now, can you answer me this? Is it possible for an object instance to be the object itself?

MR. JOY: I would say just not generally -- in reality, no, you're never downloading instances; you're downloading, basically, a set of object templates, and you create instances from those templates. It's the whole object-oriented program, you know, model of -- you have a set of defined objects, and you have a functionality on a user computer that's capable of creating instances of those objects on demand.

JUDGE LORIN: I read your specification, and I think it defines instances as copies of the object.

MR. JOY: Okay. The object -- I would say that someone of ordinary skill in the art would understand that to be an object template.

"Object" is probably not the -- the perfect term to use, but I'd say that people, you know, in the programming area would understand that to mean creating an instance from an object template.

JUDGE LORIN: Okay. Okay, Counsel. Okay. Thank you.

JUDGE FETTING: I guess my question is what about the somewhat unique case of where the -- we're talking about the program itself as the

object? Doesn't -- doesn't Archibald at least monitor the -- the rather unique object of the program itself?

MR. JOY: I don't -- I don't believe that would match up with the definition of "object" as used in this thing. You have to -- you have to create an instance from an object class, and I don't believe that that would fit. I -- I -- I don't profess to be an expert in -- in software, but I do -- I think I know enough, probably enough to be dangerous, that I don't think that's the way it would work. You never have -- you never have just a whole application being an object. I -- I have never known that to be the case, but I could stand corrected by a better programmer than me. It's been too long. I graduated from undergraduate 20-some years ago.

JUDGE FETTING: I'm in the same boat.

MR. JOY: I'm having trouble enough following what they're doing to the law, so --

JUDGE FETTING: Right. But object class is just a label that -- that the programmers put on the things. Once -- once the program has been written, the object class is -- really is transparent. It's just -- it's just code that's sitting there, you know, in the disk.

When you say "downloading," all that's happening is that the processor is taking the instructions that tell it to replicate the code and create data and -- and parts of the code. You know, where is -- where is the so-called object class at that point? It seems as though it's gone by the boards between the source program and the compiled program.

MR. JOY: Well, that's the -- I think even people who are in programming would say that there's really no -- there's, you know, two types of things that you can do in object-oriented programming. One is program the object-oriented world, or you can actually have run-time -- you know, you can execute in the object-oriented world, and I would -- I would say that this particular invention is -- is addressing the creation of instances, object instances in the run-time world.

JUDGE FETTING: Okay. Well, I guess that raises the issue, then, isn't the program itself a run-time module in that it's executed -- it's executed literally at run-time. I mean, you have to use the operating system in order to actually execute it, but --

MR. JOY: I can only say that, from the point of view of -- I don't know -- you know, it's been quite a while, and I filed -- wrote and filed this application 10 years ago. I don't want to get into the details of -- I'm sure you can look -- from the on-line -- actually, it isn't even on-line.

There's a couple of years involved. I just want to say it's been quite a while, but I -- I believe that, you know, this system is presented as a situation where you download, you know, a set of object classes, and there's templates, and I -- I am just -- I know what the system itself is.

I'm not sure exactly how it was presented in the -- the wording of the application specification, because I really haven't -- I didn't focus on -- on this particular aspect when I was reviewing my -- my original application, but the point I'm trying to make is that this is really a situation -- the

system -- the intended system out there -- the actual existing system is a situation where there's a set of object classes.

There's a -- there's a -- there's other programs which use instances created of those objects, and there's basically an object factory which creates instances of those object templates on demand in a run-time environment. I don't know if that means anything, to tell you the truth, but I'm doing the best I can here.

JUDGE FETTING: Okay. Okay. I mean, I guess I'm imagining you're talking about something like a DLL library where there --

MR. JOY: Yes, yes, definitely.

JUDGE FETTING: Okay.

MR. JOY: In fact, there's some later applications that I've filed for the same company and same group of -- not the same inventor but other inventors. It's called Orchestra, is the name of the program environment, the run-time environment, and it basically -- people receive a library of object templates that they're able to run in a run-time environment by creating instances of those objects, or also develop new ones, and it becomes a child of a existing template.

JUDGE FETTING: Okay.

JUDGE LORIN: Okay, Counsel. Are you -- are you through with your comments?

MR. JOY: Yes, I am.

JUDGE LORIN: We have -- we have no more comments, no more questions.

So, thank you, Counsel. We'll take your comments under advisement, and the hearing is now closed.

(Whereupon, at 9:57 a.m., the proceedings were concluded.)